

1 [0001] SECURE CLUSTER CONFIGURATION
2 DATA SET TRANSFER PROTOCOL
3

4 [0002] Inventors:
5 Pu Paul Zhang
6 Duc Pham
7 Tien Le Nguyen
8 Peter Tsai
9
10

11 [0003] Background of the Invention

12 [0004] Field of the Invention:

13 [0005] The present invention is generally related to the coordinated
14 control of server systems utilized to provide network services and, in particular, to
15 techniques for securely coordinating and distributing configuration data among
16 a cluster of network servers and coordinating the implementation of the
17 configuration data with respect to the cluster systems and host computers systems
18 that request execution of network services.
19

20 [0006] Description of the Related Art:

21 [0007] The concept and need for load-balancing arises in a number
22 of different computing circumstances, most often as a requirement for increasing
23 the reliability and scalability of information serving systems. Particularly in the
24 area of networked computing, load-balancing is commonly encountered as a
25 means for efficiently utilizing, in parallel, a large number of information server
26 systems to respond to various processing requests including requests for data from

1 typically remote client computer systems. A logically parallel arrangement of
2 servers adds an intrinsic redundant capability while permitting performance to be
3 scaled linearly, at least theoretically, through the addition of further servers.
4 Efficient distribution of requests and moreover the resulting load then becomes an
5 essential requirement to fully utilizing the paralleled cluster of servers and
6 maximizing performance.

7 [0008] Many different systems have been proposed and variously
8 implemented to perform load-balancing with distinctions typically dependent on
9 the particularities of the load-balancing application. Chung et al. (US Patent
10 6,470,389) describes the use of a server-side central dispatcher that arbitrates the
11 selection of servers to respond to client domain name service (DNS) requests.
12 Clients direct requests to a defined static DNS cluster-server address that
13 corresponds to the central dispatcher. Each request is then redirected by the
14 dispatcher to an available server that can then return the requested information
15 directly to the client. Since each of the DNS requests are atomic and require well-
16 defined server operations, actual load is presumed to be a function of the rate of
17 requests made to each server. The dispatcher therefore implements just a basic
18 hashing function to distribute requests uniformly to the servers participating in the
19 DNS cluster.

20 [0009] The use of a centralized dispatcher for load-balancing control is
21 architecturally problematic. Since all requests flow through the dispatcher, there
22 is an immediate exposure to a single-point failure stopping the entire operation
23 of the server cluster. Further, there is no direct way to scale the performance of
24 the dispatcher. To handle larger request loads or more complex load-balancing

1 algorithms, the dispatcher must be replaced with higher performance hardware
2 at substantially higher cost.

3 [0010] As an alternative, Chung et al. proposes broadcasting all client
4 requests to all servers within the DNS cluster, thereby obviating the need for a
5 centralized dispatcher. The servers implement mutually exclusive hash functions
6 in individualized broadcast request filter routines to select requests for unique local
7 response. This approach has the unfortunate consequence of requiring each
8 server to initially process, to some degree, each DNS request, reducing the
9 effective level of server performance. Further, the selection of requests to service
10 based on a hash of the requesting client address in effect locks individual DNS
11 servers to statically defined groups of clients. The assumption of equal load
12 distribution will therefore be statistically valid, if at all, only over large numbers of
13 requests. The static nature of the policy filter routines also means that all of the
14 routines must be changed every time a server is added or removed from the
15 cluster to ensure that all requests will be selected by a unique server. Given that
16 in a large server cluster, individual server failures are not uncommon and indeed
17 must be planned for, administrative maintenance of such a cluster is likely difficult
18 if not impractical.

19 [0011] Other techniques have been advanced to load-balance networks of
20 servers under various operating conditions. Perhaps the most prevalent load-
21 balancing techniques take the approach of implementing a background or out-of-
22 channel load monitor that accumulates the information necessary to determine
23 when and where to shift resources among the servers dynamically in response to
24 the actual requests being received. For example, Jorden et al. (US Patent
25 6,438,652) describes a cluster of network proxy cache servers where each server

1 further operates as a second level proxy cache for all of the other servers within
2 the cluster. A background load monitor observes the server cluster for repeated
3 second level cache requests for particular content objects. Excessive requests for
4 the same content satisfied from the same second level cache is considered an
5 indication that the responding server is overburdened. Based on a balancing of
6 the direct or first level cache request frequency being served by a server and the
7 second level cache request frequency, the load monitor determines whether to
8 copy the content object to one or more other caches, thereby spreading the
9 second level cache work-load for broadly and repeatedly requested content
10 objects.

11 [0012] Where resources, such as simple content objects, cannot be readily
12 shifted to effect load-balancing, alternate approaches have been developed that
13 characteristically operate by selectively transferring requests, typically represented
14 as tasks or processes, to other servers within a cluster network of servers. Since
15 a centralized load-balancing controller is preferably to be avoided, each server
16 is required to implement a monitoring and communications mechanism to
17 determine which other server can accommodate a request and then actually
18 provide for the corresponding request transfer. The process transfer aspect of the
19 mechanism is often implementation specific in that the mechanism will be highly
20 dependent on the particular nature of the task to transfer and range in complexity
21 from a transfer of a discrete data packet representing the specification of a task
22 to the collection and transport of the entire state of an actively executing process.
23 Conversely, the related conventional load monitoring mechanisms can be
24 generally categorized as source or target oriented. Source oriented servers
25 actively monitor the load status of target servers by actively inquiring of and

1 retrieving the load status of at least some subset of target servers within the
2 cluster. Target oriented load monitoring operates on a publication principle
3 where individual target servers broadcast load status information reflecting, at a
4 minimum, a capacity to receive a task transfer.

5 [0013] In general, the source and target sharing of load status information
6 is performed at intervals to allow other servers within the cluster to obtain on
7 demand or aggregate over time some dynamic representation of the available
8 load capacity of the server cluster. For large server clusters, however, the load
9 determination operations are often restricted to local or server relative network
10 neighborhoods to minimize the number of discrete communications operations
11 imposed on the server cluster as a whole. The trade-off is that more distant server
12 load values must propagate through the network over time and, consequently,
13 result in inaccurate loading reports that lead to uneven distribution of load.

14 [0014] A related problem is described in Allon et al. (US Patent 5,539,883).
15 Server load values, collected into a server cluster load vector, are incrementally
16 requested or advertized by the various servers of the server cluster. Before a
17 server transfers a local copy of the vector, the load values for the server are
18 updated in the vector. Servers receiving the updated vector in turn update the
19 server local copy of the vector with the received load values based on defined
20 rules. Consequently, the redistribution of load values for some given
21 neighborhood may expose an initially lightly loaded server to a protracted high
22 demand for services. The resulting task overload and consequential refusal of
23 service will last at least until a new load vector reflecting the higher server load
24 values circulates among a sufficient number of the servers to properly reflect the
25 load. To alleviate this problem, Allon et al. further describes a tree-structured

1 distribution pattern for load value information as part of the load-balancing
2 mechanism. Based on the tree-structured transfer of load information, low load
3 values, identifying lightly loaded servers, are aged through distribution to preclude
4 lightly loaded servers from being flooded with task transfers.

5 [0015] Whether source or target originated, load-balancing based on the
6 periodic sharing of load information between the servers of the server cluster
7 operates on the fundamental assumption that the load information is reliable as
8 finally delivered. Task transfer rejections are conventionally treated as
9 fundamental failures and, while often recoverable, require extensive exception
10 processing. Consequently, the performance of individual servers may tend to
11 degrade significantly under progressively increasing load, rather than stabilize, as
12 increasing numbers of task transfer recovery and retries operations are required
13 to ultimately achieve a balanced load distribution.

14 [0016] In circumstances where high load conditions are normally incurred,
15 specialized network protocols have been developed to accelerate the exchange
16 and certainty of loading information. Routers and other switch devices are often
17 clustered in various configurations to share network traffic load. A linking network
18 protocol is provided to provide fail-over monitoring in local redundant router
19 configurations and to share load information between both local and remote
20 routers. Current load information, among other shared information, is
21 propagated at high frequency between devices to continuously reflect the
22 individual load status of the clustered devices. As described in Bare (US Patent
23 6,493,318) for example, protocol data packets can be richly detailed with
24 information to define and manage the propagation of the load information and
25 to further detail the load status of individual devices within the cluster. Sequence

1 numbers, hop counts, and various flag-bits are used in support of spanning tree-
2 type information distribution algorithms to control protocol packet propagation
3 and prevent loop-backs. The published load values are defined in terms of
4 internal throughput rate and latency cost, which allows other clustered routers a
5 more refined basis for determining preferred routing paths. While effective, the
6 custom protocol utilized by the devices described in Bare essentially requires that
7 substantial parts of the load-balancing protocol be implemented in specialized,
8 high-speed hardware, such as network processors. The efficient handling of such
9 protocols is therefore limited to specialized, not general purpose computer
10 systems.

11 [0017] Ballard (US Patent 6,078,960) describes a client/server system
12 architecture that, among other features, effects a client-directed load-balanced
13 use of a server network. For circumstances where the various server computer
14 systems available for use by client computer systems may be provided by
15 independent service providers and where use of the different servers may involve
16 different cost structures, Ballard describes a client-based approach for selectively
17 distributing load from the clients to distinct individual servers within the server
18 network. By implementing client-based load-balancing, the client computer
19 systems in Ballard are essentially independent of the service provider server
20 network implementation.

21 [0018] To implement the Ballard load-balancing system, each client
22 computer system is provided with a server identification list from which servers are
23 progressively selected to receive client requests. The list specifies load control
24 parameters, such as the percentage load and maximum frequency of client
25 requests that are to be issued, for each server identified in the list. Server loads

1 are only roughly estimated by the clients based on the connection time necessary
2 for a request to complete or the amount of data transferred in response to a
3 request. Client requests are then issued by the individual clients to the servers
4 selected as necessary to statistically conform to the load-balancing profile defined
5 by the load control parameters. While the server identification list and included
6 load control parameters are static as held by a client, the individual clients may
7 nonetheless retrieve new server identification lists at various intervals from
8 dedicated storage locations on the servers. Updated server identification lists are
9 distributed to the servers as needed under the manual direction of an
10 administrator. Updating of the server identification lists allows an administrator
11 to manually adjust the load-balance profiles as needed due to changing client
12 requirements and to accommodate the addition and removal of servers from the
13 network.

14 [0019] The static nature of the server identification lists makes the client-
15 based load-balancing operation of the Ballard system fundamentally
16 unresponsive to the actual operation of the server network. While specific server
17 loading can be estimated by the various clients, only complete failures to respond
18 to client requests are detectable and then handled only by excluding a non-
19 responsive server from further participation in servicing client requests.
20 Consequently, under dynamically varying loading conditions, the one sided load-
21 balancing performed by the clients can seriously misapprehend the actual loading
22 of the server network and further exclude servers from participation at least until
23 re-enabled through manual administrative intervention. Such blind exclusion of
24 a server from the server network only increases the load on the remaining servers
25 and the likelihood that other servers will, in turn, be excluded from the server

1 network. Constant manual administrative monitoring of the active server network,
2 including the manual updating of server identification lists to re-enable servers
3 and to adjust the collective client balancing of load on the server network, is
4 therefore required. Such administrative maintenance is quite slow, at least relative
5 to how quickly users will perceive occasions of poor performance, and costly to
6 the point of operational impracticality.

7 [0020] From the forgoing discussion, it is evident that an improved system
8 and methods for cooperatively load-balancing a cluster of servers is needed.
9 There is also a further need, not even discussed in the prior art, for cooperatively
10 managing the configuration of a server cluster, not only with respect to the
11 interoperation of the servers as part of the cluster, but further as a server cluster
12 providing a composite service to external client computer systems. Also,
13 unaddressed is any need for security over the information exchanged between the
14 servers within a cluster. As clustered systems become more widely used for
15 security sensitive purposes, diversion of any portion of the cluster operation
16 through the interception of shared information or introduction of a compromised
17 server into the cluster represents an unacceptable risk.

18
19
20 [0021] Summary of the Invention

21 [0022] Thus, a general purpose of the present invention is to provide an
22 efficient system and methods of securely coordinating and distributing
23 configuration data among a cluster of network servers to effectively provide a
24 secure system of managing the common configuration of a scalable network
25 service.

1 [0023] This is achieved in the present invention by securely managing
2 communications between server computer systems within a cluster to maintain
3 control over the identification of configuration updates and the distribution of
4 updated configuration data sets. Configuration status messages are routinely
5 exchanged among the servers over a communications network. Each status
6 message identifies any change in the local configuration of a servers and, further,
7 includes encrypted validation data. Each of the servers stores respective
8 configuration data including respective sets of data identifying the servers known
9 to the respective servers as participating in the cluster. Each status message, as
10 received, is validating against the respective configuration data stored by the
11 receiving server. A status message is determined valid when originating from a
12 server as known by the receiving server, as determined from the configuration
13 data held by the receiving server. Where a validated originating server identifies
14 updated configuration data, the receiving server equivalently modifies the locally
15 held configuration data. The configuration of the cluster thus converges on the
16 updated configuration.

17 [0024] Thus, an advantage of the present invention is that acceptance of
18 notice of any update to the configuration data and, further, the acceptance of any
19 subsequently received updated configuration data set is constrained to the set of
20 servers that are mutually known to one another. A receiving server will only
21 accept as valid a message that originates from a server that is already known to
22 the receiving server. Thus, the cluster is a securely closed set of server systems.

23 [0025] Another advantage of the present invention is that, since only light-
24 weight status messages are routinely transmitted among the servers of the cluster,
25 there is minimal processing overhead imposed on the servers to maintain a

1 consistent overall configuration. Updated configuration data sets are transmitted
2 only on demand and generally only occurring after an administrative update is
3 performed.

4 [0026] A further advantage of the present invention is that status messages
5 can serve to both identify the availability of updated configuration sets and to
6 subsequently coordinate the mutual installation of the new configuration data,
7 thereby ensuring consistent operation of the cluster. Configuration version control
8 is also asserted against the host computers to ensure consistent operation.

9 [0027] Still another advantage of the present invention is that both status
10 messages and the transmitted configuration data sets are validated to ensure that
11 they originate from a known participant of the cluster. Receipt validation ensures
12 that rogues and trojans cannot infiltrate the cluster.

13 [0028] Yet another advantage of the present invention is that updated
14 configuration data sets, as encrypted for transmission on demand between servers
15 of the cluster, are further structured to ensure decryption only by a server of the
16 cluster already known to the server that prepares the encrypted, updated
17 configuration data set for transmission.

18
19
20 [0029] Brief Description of the Drawings

21 [0030] Figure 1A is a network diagram illustrating a system environment
22 within which host computer systems directly access network services provided by
23 a server cluster in accordance with a preferred embodiment of the present
24 invention.

1 [0031] Figure 1B is a network diagram illustrating a system environment
2 within which a preferred core network gateway embodiment of the present
3 invention is implemented.

4 [0032] Figure 2 is a detailed block diagram showing the network
5 interconnection between an array of hosts and a cluster of security processor
6 servers constructed in accordance with a preferred embodiment of the present
7 invention.

8 [0033] Figure 3 is a detailed block diagram of a security processor server
9 as constructed in accordance with a preferred embodiment of the present
10 invention.

11 [0034] Figure 4 is a block diagram of a policy enforcement module control
12 process as implemented in a host computer system in accordance with a preferred
13 embodiment of the present invention.

14 [0035] Figure 5 is a simplified block diagram of a security processor server
15 illustrating the load-balancing and policy update functions shared by a server
16 cluster service provider in accordance with a preferred embodiment of the present
17 invention.

18 [0036] Figure 6 is a flow diagram of a transaction process cooperatively
19 performed between a policy enforcement module process and a selected cluster
20 server in accordance with a preferred embodiment of the present invention.

21 [0037] Figure 7A is a flow diagram of a secure cluster server policy update
22 process as performed between the members of a server cluster in accordance with
23 a preferred embodiment of the present invention.

1 [0038] Figure 7B is a block illustration of a secure cluster server policy
2 synchronization message as defined in accordance with a preferred embodiment
3 of the present invention.

4 [0039] Figure 7C is a block illustration of a secure cluster server policy data
5 set transfer message data structure as defined in accordance with a preferred
6 embodiment of the present invention.

7 [0040] Figure 8 is a flow diagram of a process to regenerate a secure
8 cluster server policy data set transfer message in accordance with a preferred
9 embodiment of the present invention.

10 [0041] Figure 9 is a flow diagram illustrating an extended transaction
11 process performed by a host policy enforcement process to account for a version
12 change in the reported secure cluster server policy data set of a cluster server in
13 accordance with a preferred embodiment of the present invention.

14

15

16 [0042] Detailed Description of the Invention

17 [0043] While system architectures have generally followed a client/server
18 paradigm, actual implementations are typically complex and encompass a wide
19 variety of layered network assets. Although architectural generalities are difficult,
20 in all there are fundamentally common requirements of reliability, scalability, and
21 security. As recognized in connection with the present invention, a specific
22 requirement for security commonly exists for at least the core assets, including the
23 server systems and data, of a networked computer system enterprise. The present
24 invention provides for a system and methods of providing a cluster of servers that
25 provide a security service to a variety of hosts established within an enterprise

1 without degrading access to the core assets while maximizing, through efficient
2 load balancing, the utilization of the security server cluster. Those of skill in the
3 art will appreciate that the present invention, while particularly applicable to the
4 implementation of a core network security service, provides fundamentally enables
5 the efficient, load-balanced utilization of a server cluster and, further, enables the
6 efficient and secure administration of the server cluster. As will also be
7 appreciated, in the following detailed description of the preferred embodiments
8 of the present invention, like reference numerals are used to designate like parts
9 as depicted in one or more of the figures.

10 [0044] A basic and preferred system embodiment 10 of the present
11 invention is shown in Figure 1A. Any number of independent host computer
12 systems 12_{1-N} are redundantly connected through a high-speed switch 16 to a
13 security processor cluster 18. The connections between the host computer systems
14 12_{1-N}, the switch 16 and cluster 18 may use dedicated or shared media and may
15 extend directly or through LAN or WAN connections variously between the host
16 computer systems 12_{1-N}, the switch 16 and cluster 18. In accordance with the
17 preferred embodiments of the present invention, a policy enforcement module
18 (PEM) is implemented on and executed separately by each of the host computer
19 systems 12_{1-N}. Each PEM, as executed, is responsible for selectively routing
20 security related information to the security processor cluster 18 to discretely qualify
21 requested operations by or on behalf of the host computer systems 12_{1-N}. For the
22 preferred embodiments of the present invention, these requests represent a
23 comprehensive combination of authentication, authorization, policy-based
24 permissions and common filesystem related operations. Thus, as appropriate, file
25 data read or written with respect to a data store, generically shown as data store

1 14, is also routed through the security processor cluster 18 by the PEM executed
2 by the corresponding host computer systems 12_{1-N}. Since all of the operations of
3 the PEMs are, in turn, controlled or qualified by the security processor cluster 18,
4 various operations of the host computer systems 12_{1-N} can be securely monitored
5 and qualified.

6 [0045] An alternate enterprise system embodiment 20 of the present
7 invention implementation of the present invention is shown in Figure 1B. An
8 enterprise network system 20 may include a perimeter network 22 interconnecting
9 client computer systems 24_{1-N} through LAN or WAN connections to at least one
10 and, more typically, multiple gateway servers 26_{1-M} that provide access to a core
11 network 28. Core network assets, such as various back-end servers (not shown),
12 SAN and NAS data stores 30, are accessible by the client computer systems 24_{1-N}
13 through the gateway servers 26_{1-M} and core network 28.

14 [0046] In accordance with the preferred embodiments of the present
15 invention, the gateway servers 26_{1-M} may implement both perimeter security with
16 respect to the client computer systems 14_{1-N} and core asset security with respect
17 to the core network 28 and attached network assets 30 within the perimeter
18 established by the gateway servers 26_{1-M}. Furthermore, the gateway servers 26_{1-M}
19 may operate as application servers executing data processing programs on behalf
20 of the client computer systems 24_{1-N}. Nominally, the gateway servers 26_{1-M} are
21 provided in the direct path for the processing of network file requests directed to
22 core network assets. Consequently, the overall performance of the network
23 computer system 10 will directly depend, at least in part, on the operational
24 performance, reliability, and scalability of the gateway servers 26_{1-M}.

1 [0047] In implementing the security service of the gateway servers 26_{1-M},
2 client requests are intercepted by each of the gateway servers 26_{1-M} and redirected
3 through a switch 16 to a security processor cluster 18. The switch 16 may be a
4 high-speed router fabric where the security processor cluster 18 is local to the
5 gateway servers 26_{1-M}. Alternatively, conventional routers may be employed in a
6 redundant configuration to establish backup network connections between the
7 gateway servers 26_{1-M} and security processor cluster 18 through the switch 16.

8 [0048] For both embodiments 10, 20, shown in Figures 1A and 1B, the
9 security processor cluster 18 is preferably implemented as a parallel organized
10 array of server computer systems, each configured to provide a common network
11 service. In the preferred embodiments of the present invention, the provided
12 network service includes a firewall-based filtering of network data packets,
13 including network file data transfer requests, and the selective bidirectional
14 encryption and compression of file data, which is performed in response to
15 qualified network file requests. These network requests may originate directly with
16 the host computer systems 12_{1-N}, client computer systems 14_{1-N}, and gateway
17 servers 16_{1-M} operating as, for example, application servers or in response to
18 requests received by these systems. The detailed implementation and processes
19 carried out by the individual servers of the security processor cluster 18 are
20 described in copending applications Secure Network File Access Control System,
21 Serial Number 10/201,406, Filed July 22, 2002, Logical Access Block Processing
22 Protocol for Transparent Secure File Storage, Serial Number 10/201,409, Filed
23 July 22, 2002, Secure Network File Access Controller Implementing Access
24 Control and Auditing, Serial Number 10/201,358, Filed July 22, 2002, and
25 Secure File System Server Architecture and Methods, Serial Number 10/271,050,

1 Filed October 16, 2002, all of which are assigned to the assignee of the present
2 invention and hereby expressly incorporated by reference.

3 [0049] The interoperation 40 of an array of host computers 12_{1-x} and the
4 security processor cluster 18 is shown in greater detail in Figure 2. For the
5 preferred embodiments of the present invention, the host computers 12_{1-x} are
6 otherwise conventional computer systems variously operating as ordinary host
7 computer systems, whether specifically tasked as client computer systems, network
8 proxies, application servers, and database servers. A PEM component 42_{1-x} is
9 preferably installed and executed on each of the host computers 12_{1-x} to
10 functionally intercept and selectively process network requests directed to any local
11 and core data stores 14, 30. In summary, the PEM components 42_{1-x} selectively
12 forward specific requests in individual transactions to target servers 44_{1-y} within
13 the security processor cluster 18 for policy evaluation and, as appropriate, further
14 servicing to enable completion of the network requests. In forwarding the
15 requests, the PEM components 42_{1-x} preferably operate autonomously.
16 Information regarding the occurrence of a request or the selection of a target
17 server 44_{1-y} within the security processor cluster 18 is not required to be shared
18 between the PEM components 42_{1-x} , particularly on any time-critical basis.
19 Indeed, the PEM components 42_{1-x} have no required notice of the presence or
20 operation of other host computers 12_{1-x} throughout operation of the PEM
21 components 42_{1-x} with respect to the security processor cluster 18.

22 [0050] Preferably, each PEM component 42_{1-x} is initially provided with a list
23 identification of the individual target servers 44_{1-y} within the security processor
24 cluster 18. In response to a network request, a PEM component 42_{1-x} selects a
25 discrete target server 44 for the processing of the request and transmits the

1 request through the IP switch 16 to the selected target server 44. Particularly
2 where the PEM component 42_{1,x} executes in response to a local client process, as
3 occurs in the case of application server and similar embodiments, session and
4 process identifier access attributes associated with the client process are collected
5 and provided with the network request. This operation of the PEM component
6 42_{1,x} is particularly autonomous in that the forwarded network request is
7 preemptively issued to a selected target server 44 with the presumption that the
8 request will be accepted and handled by the designated target server 44.

9 [0051] In accordance with the present invention, a target servers 44_{1,y} will
10 conditionally accept a network request depending on the current resources
11 available to the target server 44_{1,y} and a policy evaluation of the access attributes
12 provided with the network request. Lack of adequate processing resources or a
13 policy violation, typically reflecting a policy determined unavailability of a local or
14 core asset against which the request was issued, will result in the refusal of the
15 network request by a target server 44_{1,y}. Otherwise, the target server 44_{1,y}
16 accepts the request and performs the required network service.

17 [0052] In response to a network request, irrespective of whether the request
18 is ultimately accepted or rejected, a target server 44_{1,y} returns load and,
19 optionally, weight information as part of the response to the PEM component
20 42_{1,x} that originated the network request. The load information provides the
21 requesting PEM component 42_{1,x} with a representation of the current data
22 processing load on the target server 44_{1,y}. The weight information similarly
23 provides the requesting PEM component 42_{1,x} with a current evaluation of the
24 policy determined prioritizing weight for a particular network request, the
25 originating host 12 or gateway server 26 associated with the request, set of access

1 attributes, and the responding target server 44_{1,Y}. Preferably, over the course of
2 numerous network request transactions with the security processor cluster 18, the
3 individual PEM components 42_{1,X} will develop preference profiles for use in
4 identifying the likely best target server 44_{1,Y} to use for handling network requests
5 from specific client computer systems 12_{1,N} and gateway servers 26_{1,M}. While load
6 and weight values reported in individual transactions will age with time and may
7 further vary based on the intricacies of individual policy evaluations, the ongoing
8 active utilization of the host computer systems 12_{1,N} permits the PEM components
9 42_{1,X} to develop and maintain substantially accurate preference profiles that tend
10 to minimize the occurrence of request rejections by individual target servers 44_{1,Y}.
11 The load distribution of network requests is thereby balanced to the degree
12 necessary to maximize the acceptance rate of network request transactions.

13 [0053] As with the operation of the PEM components 42_{1,X}, the operation
14 of the target servers 44_{1,Y} are essentially autonomous with respect to the receipt
15 and processing of individual network requests. In accordance with the preferred
16 embodiments of the present invention, load information is not required to be
17 shared between the target servers 44_{1,Y} within the cluster 18, particularly in the
18 critical time path of responding to network requests. Preferably, the target servers
19 44_{1,Y} uniformly operate to receive any network requests presented and, in
20 acknowledgment of the presented request, identify whether the request is
21 accepted, provide load and optional weight information, and specify at least
22 implicitly the reason for rejecting the request.

23 [0054] While not particularly provided to share load information, a
24 communications link between the individual target servers 44_{1,Y} within the security
25 processor cluster 18 is preferably provided. In the preferred embodiments of the

1 present invention, a cluster local area network 46 is established in the preferred
2 embodiments to allow communication of select cluster management information,
3 specifically presence, configuration, and policy information, to be securely shared
4 among the target servers 44_{1-Y}. The cluster local area network 46
5 communications are protected by using secure sockets layer (SSL) connections and
6 further by use of secure proprietary protocols for the transmission of the
7 management information. Thus, while a separate, physically secure cluster local
8 area network 46 is preferred, the cluster management information may be routed
9 over shared physical networks as necessary to interconnect the target servers 44_{1-Y}
10 of the security processor cluster 18.

11 [0055] Preferably, presence information is transmitted by a broadcast
12 protocol periodically identifying, using encrypted identifiers, the participating
13 target servers 44_{1-Y} of the security processor cluster 18. The security information
14 is preferably transmitted using a lightweight protocol that operates to ensure the
15 integrity of the security processor cluster 18 by precluding rogue or Trojan devices
16 from joining the cluster 18 or compromising the secure configuration of the target
17 servers 44_{1-Y}. A set of configuration policy information is communicated using an
18 additional lightweight protocol that supports controlled propagation of
19 configuration information, including a synchronous update of the policy rules
20 utilized by the individual target servers 44_{1-Y} within the security processor cluster
21 18. Given that the presence information is transmitted at a low frequency relative
22 to the nominal rate of network request processing, and the security and
23 configuration policy information protocols execute only on the administrative
24 reconfiguration of the security processor cluster 18, such as through the addition
25 of target servers 44_{1-Y} and entry of administrative updates to the policy rule sets,

1 the processing overhead imposed on the individual target servers 44_{1,y} to support
2 intra-cluster communications is negligible and independent of the cluster loading.

3 [0056] A block diagram and flow representation of the software
4 architecture 50 utilized in a preferred embodiment of the present invention is
5 shown in Figure 3. Generally inbound network request transactions are processed
6 through a hardware-based network interface controller that supports routeable
7 communications sessions through the switch 16. These inbound transactions are
8 processed through a first network interface 52, a protocol processor 54, and a
9 second network interface 54, resulting in outbound transactions redirected
10 through the host computers 12_{1,x} to local and core data processing and storage
11 assets 14, 30. The same, separate, or multiple redundant hardware network
12 interface controllers can be implemented in each target server 44_{1,y} and
13 correspondingly used to carry the inbound and outbound transactions through the
14 switch 16.

15 [0057] Network request data packets variously received by a target server
16 44 from PEM components 42_{1,x}, each operating to initiate corresponding network
17 transactions against local and core network assets 14, 30, are processed through
18 the protocol processor 54 to initially extract selected network and application data
19 packet control information. Preferably, this control information is wrapped in a
20 conventional TCP data packet by the originating PEM component 42_{1,x} for
21 conventional routed transfer to the target server 44_{1,y}. Alternately, the control
22 information can be encoded as a proprietary RPC data packet. The extracted
23 network control information includes the TCP, IP, and similar networking protocol
24 layer information, while the extracted application information includes access
25 attributes generated or determined by operation of the originating PEM

1 component 42_{1,x} with respect to the particular client processes and context within
2 which the network request is generated. In the preferred embodiments of the
3 present invention, the application information is a collection of access attributes
4 that directly or indirectly identifies the originating host computer, user and
5 domain, application signature or security credentials, and client session and
6 process identifiers, as available, for the host computer 12_{1,N} that originates the
7 network request. The application information preferably further identifies, as
8 available, the status or level of authentication performed to verify the user.
9 Preferably, a PEM component 42_{1,x} automatically collects the application
10 information into a defined data structure that is then encapsulated as a TCP
11 network data packet for transmission to a target server 44_{1,y}.

12 [0058] Preferably, the network information exposed by operation of the
13 protocol processor 54 is provided to a transaction control processor 58 and both
14 the network and application control information is provided to a policy parser 60.
15 The transaction control processor 58 operates as a state machine that controls the
16 processing of network data packets through the protocol processor 54 and further
17 coordinates the operation of the policy parser in receiving and evaluating the
18 network and application information. The transaction control processor 58 state
19 machine operation controls the detailed examination of individual network data
20 packets to locate the network and application control information and, in
21 accordance with the preferred embodiments of the present invention, selectively
22 control the encryption and compression processing of an enclosed data payload.
23 Network transaction state is also maintained through operation of the transaction
24 control processor 58 state machine. Specifically, the sequences of the network
25 data packets exchanged to implement network file data read and write

1 operations, and other similar transactional operations, are tracked as necessary
2 to maintain the integrity of the transactions while being processed through the
3 protocol processor 54.

4 [0059] In evaluating a network data packet identified by the transaction
5 control processor 58 as an initial network request, the policy parser 60 examines
6 selected elements of the available network and application control information.
7 The policy parser 60 is preferably implemented as a rule-based evaluation engine
8 operating against a configuration policy/key data set stored in a policy/key store
9 62. The rules evaluation preferably implements decision tree logic to determine
10 the level of host computer 12_{1-N} authentication required to enable processing the
11 network file request represented by the network file data packet received, whether
12 that level of authentication has been met, whether the user of a request initiating
13 host computer 12_{1-N} is authorized to access the requested core network assets,
14 and further whether the process and access attributes provided with the network
15 request are adequate to enable access to the specific local or core network
16 resource 14, 30 identified in the network request.

17 [0060] In a preferred embodiment of the present invention, the decision
18 tree logic evaluated in response to a network request to access file data considers
19 user authentication status, user access authorization, and access permissions.
20 Authentication of the user is considered relative to a minimum required
21 authentication level defined in the configuration policy/key data set against a
22 combination of the identified network request core network asset, mount point,
23 target directory and file specification. Authorization of the user against the
24 configuration policy/key data set is considered relative to a combination of the
25 particular network file request, user name and domain, client IP, and client session

1 and client process identifier access attributes. Finally, access permissions are
2 determined by evaluating the user name and domains, mount point, target
3 directory and file specification access attributes with correspondingly specified
4 read/modify/write permission data and other available file related function and
5 access permission constraints as specified in the configuration policy/key data set.

6 [0061] Where PEM components 42_{1-x} function as filesystem proxies, useful
7 to map and redirect filesystem requests for virtually specified data stores to
8 particular local and core network file system data stores 14, 30, data is also
9 stored in the policy/key store 62 to define the set identity of virtual file system
10 mount points accessible to host computer systems 12_{1-N} and the mapping of
11 virtual mount points to real mount points. The policy data can also variously
12 define permitted host computer source IP ranges, whether application
13 authentication is to be enforced as a prerequisite for client access, a limited,
14 permitted set of authenticated digital signatures of authorized applications,
15 whether user session authentication extends to spawned processes or processes
16 with different user name and domain specifications, and other attribute data that
17 can be used to match or otherwise discriminate, in operation of the policy parser
18 60, against application information that can be marshaled on demand by the
19 PEM components 42_{1-x} and network information.

20 [0062] In the preferred embodiments of the present invention, encryption
21 keys are also stored in the policy/key store 62. Preferably, individual encryption
22 keys, as well as applicable compression specifications, are maintained in a
23 logically hierarchical policy set rule structure parseable as a decision tree. Each
24 policy rule provides an specification of some combination of network and
25 application attributes, including the access attributed defined combination of

1 mount point, target directory and file specification, by which permissions
2 constraints on the further processing of the corresponding request can be
3 discriminated. Based on a pending request, a corresponding encryption key is
4 parsed by operation of the policy parser 60 from the policy rule set as needed by
5 the transaction control processor 58 to support the encryption and decryption
6 operations implemented by the protocol processor subject. For the preferred
7 embodiments of the present invention, policy rules and related key data are stored
8 in a hash table permitting rapid evaluation against the network and application
9 information.

10 [0063] Manual administration of the policy data set data is performed
11 through an administration interface 64, preferably accessed over a private
12 network and through a dedicated administration network interface 66. Updates
13 to the policy data set are preferably exchanged autonomously among the target
14 servers 44_{1-y} of the security processor cluster 18 through the cluster network 46
15 accessible through a separate cluster network interface 68. A cluster policy
16 protocol controller 70 implements the secure protocols for handling presence
17 broadcast messages, ensuring the security of the cluster 46 communications, and
18 exchanging updates to the configuration policy/key data set data.

19 [0064] On receipt of a network request, the transaction control processor
20 58 determines whether to accept or reject the network request dependent on the
21 evaluation performed by the policy parser 60 and the current processing load
22 values determined for the target server 44. A policy parser 60 based rejection will
23 occur where the request fails authentication, authorization, or permissions policy
24 evaluation. For the initially preferred embodiments of the present invention,
25 rejections are not issued for requests received in excess of the current processing

1 capacity of a target server 44. Received requests are buffered and processed in
2 order of receipt with an acceptable increase in the request response latency. The
3 load value immediately returned in response to a request that is buffered will
4 effectively redirect subsequent network requests from the host computers 12_{1-N} to
5 other target servers 44_{1-Y}. Alternately, any returned load value can be biased
6 upward by a small amount to minimize the receipt of network requests that are
7 actually in excess of the current processing capacity of a target server 44. In an
8 alternate embodiment of the present invention, an actual rejection of a network
9 request may be issued by a target server 44_{1-Y} to expressly preclude exceeding the
10 processing capacity of a target server 44_{1-Y}. A threshold of, for example, 95%
11 load capacity can be set to define when subsequent network requests are to be
12 refused.

13 [0065] To provide the returned load value, a combined load value is
14 preferably computed based on a combination of individual load values
15 determined for the network interface controllers connected to the primary network
16 interfaces 52, 56, main processors, and hardware-based encryption/compression
17 coprocessors employed by a target server 44. This combined load value and,
18 optionally, the individual component load values are returned to the request
19 originating host computer 12_{1-N} in response to the network request. Preferably,
20 at least the combined load value is preferably projected to include handling of the
21 current network request. Depending then on the applicable load policy rules
22 governing the operation of the target server 44_{1-Y}, the response returned signals
23 either an acceptance or rejection of the current network request.

24 [0066] In combination with authorization, authentication and permissions
25 evaluation against the network request, the policy parser 60 optionally determines

1 a policy set weighting value for the current transaction, preferably irrespective of
2 whether the network request is to be rejected. This policy determined weighting
3 value represents a numerically-based representation of the appropriateness for
4 use of a particular target server 44 relative to a particular a network request and
5 associated access attributes. For a preferred embodiment of the present
6 invention, a relative low value in a normalized range of 1 to 100, indicating
7 preferred use, is associated with desired combinations of acceptable network and
8 application information. Higher values are returned to identify generally backup
9 or alternative acceptable use. A preclusive value, defined as any value above a
10 defined threshold such as 90, is returned as an implicit signal to a PEM
11 component 42_{1,x} that corresponding network requests are not to be directed to the
12 specific target server 44 except under exigent circumstances.

13 [0067] In response to a network request, a target server 44 returns the reply
14 network data packet including the optional policy determined weighting value, the
15 set of one or more load values, and an identifier indicating the acceptance or
16 rejection of the network request. In accordance with the preferred embodiments
17 of the present invention, the reply network data packet may further specify
18 whether subsequent data packet transfers within the current transaction need be
19 transferred through the security processor cluster 18. Nominally, the data packets
20 of an entire transaction are routed through a corresponding target server 44 to
21 allow for encryption and compression processing. However, where the underlying
22 transported file data is not encrypted or compressed, or where any such
23 encryption or compression is not to be modified, or where the network request
24 does not involve a file data transfer, the current transaction transfer of data need
25 not route the balance of the transaction data packets through the security

1 processor cluster 18. Thus, once the network request of the current transaction
2 has been evaluated and approved by the policy parser 60 of a target server 44,
3 and an acceptance reply packet returned to the host computer 12_{1-N}, the
4 corresponding PEM component 42_{1-x} can selectively bypass use of the security
5 processor cluster 18 for the completion of the current transaction.

6 [0068] An exemplary representation of a PEM component 42, as executed,
7 is shown 80 in Figure 4. A PEM control layer 82, executed to implement the
8 control function of the PEM component 42, is preferably installed on a host system
9 12 as a kernel component under the operating system virtual file system switch or
10 equivalent operating system control structure. In addition to supporting a
11 conventional virtual file system switch interface to the operating system kernel, the
12 PEM control layer 82 preferably implements some combination of a native or
13 network file system or an interface equivalent to the operating system virtual file
14 system switch interface through which to support internal or operating system
15 provided file systems 84. Externally provided file systems 84 preferably include
16 block-oriented interfaces enabling connection to direct access (DAS) and storage
17 network (SAN) data storage assets and file-oriented interfaces permitting access
18 to network-attached storage (NAS) network data storage assets.

19 [0069] The PEM control layer 82 preferably also implements an operating
20 system interface that allows the PEM control layer 82 to obtain the hostname or
21 other unique identifier of the host computer system 12, the source session and
22 process identifiers corresponding to the process originating a network file request
23 as received through the virtual file system switch, and any authentication
24 information associated with the user name and domain for the process originating
25 the network file request. In the preferred embodiments of the present invention,

1 these access attributes and the network file request as received by the PEM control
2 layer 82 are placed in a data structure that is wrapped by a conventional TCP
3 data packet. This effectively proprietary TCP data packet is then transmitted
4 through the IP switch 16 to present the network request to a selected target server
5 44. Alternately, a conventional RPC structure could be used in place of the
6 proprietary data structure.

7 [0070] The selection of the target server 44 is performed by the PEM control
8 layer 82 based on configuration and dynamically collected performance
9 information. A security processor IP address list 86 provides the necessary
10 configuration information to identify each of the target servers 44_{1,Y} within the
11 security processor cluster 18. The IP address list 86 can be provided manually
12 through a static initialization of the PEM component 42 or, preferably, is retrieved
13 as part of an initial configuration data set on an initial execution of the PEM
14 control layer 82 from a designated or default target server 44_{1,Y} of the security
15 processor cluster 18. In the preferred embodiment of the present invention, each
16 PEM component 42_{1,X}, in initial execution, implements an authentication
17 transaction against the security processor cluster 18 through which the integrity
18 of the executing PEM control layer 82 is verified and the initial configuration data,
19 including an IP address list 86, is provided to the PEM component 42_{1,X}.

20 [0071] Dynamic information, such as the server load and weight values, is
21 progressively collected by an executing PEM component 42_{1,X} into a SP
22 loads/weights table 88. The load values are timestamped and indexed relative
23 to the reporting target server 44. The weight values are similarly timestamped
24 and indexed. For an initial preferred embodiment, PEM component 42_{1,X} utilizes
25 a round-robin target server 44_{1,Y} selection algorithm, where selection of a next

1 target server 44_{1,Y} occurs whenever the loading of a current target server 44_{1,Y}
2 reaches 100%. Alternately, the load and weight values may be further inversely
3 indexed by any available combination of access attributes including requesting
4 host identifier, user name, domain, session and process identifiers, application
5 identifiers, network file operation requested, core network asset reference, and
6 any mount point, target directory and file specification. Using a hierarchical
7 nearest match algorithm, this stored dynamic information allows a PEM
8 component 42_{1,X} to rapidly establish an ordered list several target servers 44_{1,Y}
9 that are both least loaded and most likely to accept a particular network request.
10 Should the first identified target server 44_{1,Y} reject the request, the next listed target
11 server 44_{1,Y} is tried.

12 [0072] A network latency table 90 is preferably utilized to store dynamic
13 evaluations of network conditions between the PEM control layer 82 and each of
14 the target servers 44_{1,Y}. Minimally, the network latency table 90 is used to identify
15 those target servers 44_{1,Y} that no longer respond to network requests or are
16 otherwise deemed inaccessible. Such unavailable target servers 44_{1,Y} are
17 automatically excluded from the target servers selection process performed by the
18 PEM control layer 82. The network latency table 90 may also be utilized to store
19 timestamped values representing the response latency times and communications
20 cost of the various target servers 44_{1,Y}. These values may be evaluated in
21 conjunction with the weight values as part of the process of determining and
22 ordering of the target servers 44_{1,Y} for receipt of new network requests.

23 [0073] Finally, a preferences table 92 may be implemented to provide a
24 default traffic shaping profile individualized for the PEM component 42_{1,X}. For
25 an alternate embodiment of the present invention, a preferences profile may be

1 assigned to each of the PEM components $42_{1,x}$ to establish a default allocation or
2 partitioning of the target servers $44_{1,y}$ within a security processor cluster 18. By
3 assigning target servers $44_{1,y}$ different preference values among the PEM
4 components $42_{1,x}$ and further evaluating these preference values in conjunction
5 with the weight values, the network traffic between the various host computers
6 $12_{1,N}$ and individual target servers $44_{1,y}$ can be used to flexibly define use of
7 particular target servers $44_{1,y}$. As with the IP address list 86, the contents of the
8 preferences table may be provided by manual initialization of the PEM control
9 layer 82 or retrieved as configuration data from the security processor cluster 18.

10 [0074] A preferred hardware server system 100 for the target servers $44_{1,y}$
11 is shown in Figure 5. In the preferred embodiments of the present invention, the
12 software architecture 50, as shown in Figure 3, is substantially executed by one
13 or more main processors 102 with support from one or more peripheral,
14 hardware-based encryption/compression engines 104. One or more primary
15 network interface controllers (NICs) 106 provide a hardware interface to the IP
16 switch 16. Other network interface controllers, such as the controller 108,
17 preferably provide separate, redundant network connections to the secure cluster
18 network 46 and to an administrator console (not shown). A heartbeat timer 110
19 preferably provides a one second interval interrupt to the main processors to
20 support maintenance operations including, in particular, the secure cluster
21 network management protocols.

22 [0075] The software architecture 50 is preferably implemented as a server
23 control program 112 loaded in and executed by the main processors 102 from
24 the main memory of the hardware server system 100. In executing the server
25 control program 112, the main processors 102 preferably perform on-demand

1 acquisition of load values for the primary network interface controller 106, main
2 processors 102, and the encryption/compression engines 104. Depending on the
3 specific hardware implementation of the network interface controller 106 and
4 encryption/compression engines 104, individual load values may be read 114
5 from corresponding hardware registers. Alternately, software-based usage
6 accumulators may be implemented through the execution of the server control
7 program 112 by the main processors 102 to track throughput use of the network
8 interface controller 106 and current percentage capacity processing utilization of
9 the encryption/compression engines 104. In the initially preferred embodiments
10 of the present invention, each of the load values represents the percentage
11 utilization of the corresponding hardware resource. The execution of the server
12 control program 112, also provides for establishment of a configuration
13 policy/key data set 116 table also preferably within the main memory of the
14 hardware server system 100 and accessible to the main processors 102. A
15 second table 118 is similarly maintained to receive an updated configuration
16 policy/key data set through operation of the secure cluster network 46 protocols.
17 [0076] Figure 6 provides a process flow diagram illustrating the load-
18 balancing operation 120A implemented by a PEM component 42_{1,x} as executed
19 on a host computer 12_{1,N} cooperatively 120B with a selected target server 44 of
20 the security processor cluster 18. On receipt 122 of a network request from a
21 client 14, typically presented through the virtual filesystem switch to the PEM
22 component 42_{1,x} as a filesystem request, the network request is evaluated by the
23 PEM component 42_{1,x} to associate available access attributes 124, including the
24 unique host identifier 126, with the network request. The PEM component 42_{1,x}

1 then selects 128 the IP address of a target server 44 from the security processor
2 cluster 18.

3 [0077] The proprietary TCP-based network request data packet is then
4 constructed to include the corresponding network request and access attributes.
5 This network request is then transmitted 130 through the IP switch 16 to the target
6 server 44. A target server response timeout period is set concurrently with the
7 transmission 130 of the network request. On the occurrence of a response
8 timeout 132, the specific target server 44 is marked in the network latency table
9 90 as down or otherwise non-responsive 134. Another target server 44 is then
10 selected 128 to receive the network request. Preferably, the selection process is
11 reexecuted subject to the unavailability of the non-responsive target server 44.
12 Alternately, the ordered succession of target servers identified upon initial receipt
13 of the network request may be transiently preserved to support retries in the
14 operation of the PEM component 42_{1,x}. Preservation of the selection list at least
15 until the corresponding network request is accepted by a target server 44 allows
16 a rejected network request to be immediately retried to the next successive target
17 server without incurring the overhead of reexecuting the target server 44 selection
18 process 128. Depending on the duration of the response timeout 132 period,
19 however, re-use of a selection list may be undesirable since any intervening
20 dynamic updates to the security processor loads and weights table 88 and
21 network latency table 90 will not be considered, potentially leading to a higher
22 rate of rejection on retries. Consequently, reexecution of the target server 44
23 selection process 128 taking into account all data in the security processor loads
24 and weights table 88 and network latency table 90 is generally preferred.

1 [0078] On receipt 120B of the TCP-based network request 136, a target
2 server 44 initially examines the network request to access to the request and
3 access attribute information. The policy parser 60 is invoked 138 to produce a
4 policy determined weight value for the request. The load values for the relevant
5 hardware components of the target server 44 are also collected. A determination
6 is then made of whether to accept or reject 140 the network request. If the access
7 rights under the policy evaluated network and application information precludes
8 the requested operation, the network request is rejected. For embodiments of the
9 present invention that do not automatically accept and buffer in all permitted
10 network requests, the network request is rejected if the current load or weight
11 values exceed the configuration established threshold load and weight limits
12 applicable to the target server 44_{1-Y}. In either event, a corresponding request
13 reply data packet is generated 142 and returned.

14 [0079] The network request reply is received 144 by the request originating
15 host computer 12_{1-N} and passed directly to the locally executing PEM component
16 42_{1-X}. The load and any returned weight values are timestamped and saved to
17 the security processor loads and weights table 88. Optionally, the network latency
18 between the target server 44 and host computer 12_{1-N}, determined from the
19 network request response data packet, is stored in the network latency table 90.
20 If the network request is rejected 148 based on insufficient access attributes 150,
21 the transaction is correspondingly completed 152 with respect to the host
22 computer 12_{1-N}. If rejected for other reasons, a next target server 44 is selected
23 128. Otherwise, the transaction confirmed by the network request reply is
24 processed through the PEM component 42_{1-X} and, as appropriate, transferring
25 network data packets to the target server 44 as necessary for data payload

1 encryption and compression processing 154. On completion of the client
2 requested network file operation 152, the network request transaction is complete
3 156.

4 [0080] The preferred secure process 160A/160B for distributing presence
5 information and responsively transferring configuration data sets, including the
6 configuration policy/key data, among the target servers 44_{1-Y} of a security
7 processor cluster 18 is generally shown in Figure 7A. In accordance with the
8 preferred embodiments of the present invention, each target server 44 transmits
9 various cluster messages on the secure cluster network 46. Preferably, a cluster
10 message 170, generally structured as shown in Figure 7B, includes a cluster
11 message header 172 that defines a message type, header version number, target
12 server 44_{1-Y} identifier or simply source IP address, sequence number,
13 authentication type, and a checksum. The cluster message header 172 further
14 includes a status value 174 and a current policy version number 146,
15 representing the assigned version number of the most current configuration and
16 configuration policy/key data set held by the target server 44 transmitting the
17 cluster message 170. The status value 174 is preferably used to define the
18 function of the cluster message. The status types include discovery of the set of
19 target servers 44_{1-Y} within the cluster, the joining, leaving and removal of target
20 servers 44_{1-Y} from the cluster, synchronization of the configuration and
21 configuration policy/key data sets held by the target servers 44_{1-Y}, and, where
22 redundant secure cluster networks 46 are available, the switch to a secondary
23 secure cluster network 46.

24 [0081] The cluster message 170, also includes a PK digest 178 that
25 contains a structured list including a secure hash of the public key, the

1 corresponding network IP, and a status field for each target server 44_{1-y} of the
2 security processor cluster 18, as known by the particular target server 44
3 originating a cluster message 170. Preferably, a secure hash algorithm, such as
4 SHA-1, is used to generate the secure public key hashes. The included status field
5 reflects the known operating state of each target server 44, including
6 synchronization in progress, synchronization done, cluster join, and cluster leave
7 states.

8 [0082] Preferably, the cluster message header 172 also includes a digitally
9 signed copy of the source target server 44 identifier as a basis for assuring the
10 validity of a received cluster message 170. Alternately, a digital signature
11 generated from the cluster message header 172 can be appended to the cluster
12 message 170. In either case, a successful decryption and comparison of the
13 source target server 44 identifier or secure hash of the cluster message header
14 172 enables a receiving target server 44 to verify that the cluster message 170 is
15 from a known source target server 44 and, where digitally signed, has not been
16 tampered with.

17 [0083] For the preferred embodiments of the present invention, the target
18 servers 44_{1-y} of a cluster 18 maintain essentially a common configuration to
19 ensure a consistent operating response to any network request made by any host
20 computer 12_{1-x}. To ensure synchronization the configuration of the target servers
21 44_{1-y}, cluster synchronization messages are periodically broadcast 160A on the
22 secure cluster network 46 by each of the target servers 44_{1-y}, preferably in
23 response to a hardware interrupt generated by the local heartbeat timer 162.
24 Each cluster synchronization message is sent 164 in a cluster message 170 with
25 a synchronization status 174 value, the current policy version level 176 of the

1 cluster 18, and the securely recognizable set of target servers 44_{1..Y} permitted to
2 participate in the security processor cluster 18, specifically from the frame of
3 reference of the target server 44 originating the cluster synchronization message
4 170.

5 [0084] Each target server 44 concurrently processes 160B broadcast cluster
6 synchronization messages 170 as received 180 from each of the other active
7 target servers 44_{1..Y} on the secure cluster network 46. As each cluster
8 synchronization message 170 is received 180 and validated as originating from
9 a target server 44 known to validly exist in the security processor cluster 18, the
10 receiving target server 44 will search 182 the digests of public keys 176 to
11 determine whether the public key of the receiving target server is contained within
12 the digest list 176. If the secure hash equivalent of the public key of a receiving
13 target server 44 is not found 184, the cluster synchronization message 170 is
14 ignored 186. Where the secure hashed public key of the receiving target server
15 44 is found in a received cluster synchronization message 170, the policy version
16 number 174 is compared to the version number of the local configuration
17 policy/key data set held by the receiving target server 44. If the policy version
18 number 174 is the same or less than that of the local configuration policy/key
19 data set, the cluster synchronization message 170 is again ignored 186.

20 [0085] Where the policy version number 174 identified in a cluster
21 synchronization message 170 is greater than that of the current active
22 configuration policy/key data set, the target server 44 issues a retrieval request
23 190, preferably using an HTTPs protocol, to the target server 44 identified within
24 the corresponding network data packet as the source of the cluster
25 synchronization message 170. The comparatively newer configuration policy/key

1 data set held by the identified source target server 44 is retrieved to update the
2 configuration policy/key data set held by the receiving target server 44. The
3 identified source target server 44 responds 192 by returning a source encrypted
4 policy set 200.

5 [0086] As generally detailed in Figure 7C, a source encrypted policy set 200
6 is preferably a defined data structure containing an index 202, a series of
7 encrypted access keys 204_{1-Z} , where Z is the number of target servers 44_{1-Y} known
8 by the identified source target server 44 to be validly participating in security
9 processor cluster 18, an encrypted configuration policy/key data set 206, and a
10 policy set digital signature 208. Since the distribution of configuration policy/key
11 data sets 206 may occur successively among the target servers 44_{1-Y} , the number
12 of valid participating target servers 44_{1-Y} may vary from the viewpoint of different
13 target servers 44_{1-Y} of the security processor cluster 18 while a new configuration
14 policy/key data set version is being distributed.

15 [0087] The index 202 preferably contains a record entry for each of the
16 known validly participating target servers 44_{1-Y} . Each record entry preferably
17 stores a secure hash of the public key and an administratively assigned identifier
18 of a corresponding target server 44_{1-Y} . By convention, the first listed record entry
19 corresponds to the source target server 44 that generated the encrypted policy set
20 200. The encrypted access keys 204_{1-Z} each contain the same triple-DES key,
21 through encrypted with the respective public keys of the known validly
22 participating target servers 44_{1-Y} . The source of the public keys used in encrypting
23 the triple-DES key is the locally held configuration policy/key data set.
24 Consequently, only those target servers 44_{1-Y} that are validly known to the target
25 server 44 that sources an encrypted policy set 200 will be able to first decrypt a

1 corresponding triple-DES encryption key 204_{i,z} and then successfully decrypt the
2 included configuration policy/key data set 206.

3 [0088] A new triple-DES key is preferably generated using a random
4 function for each policy version of an encrypted policy set 200 constructed by a
5 particular target servers 44_{i,y}. Alternately, new encrypted policy sets 200 can be
6 reconstructed, each with a different triple-DES key, in response to each HTTPs
7 request received by a particular target servers 44_{i,y}. The locally held configuration
8 policy/key data set 206 is triple-DES encrypted using the current generated triple-
9 DES key. Finally, a digital signature 208, generated based on a secure hash of
10 the index 202 and list of encrypted access keys 204_{i,z}, is appended to complete
11 the encrypted policy set 200 structure. The digital signature 208 thus ensures that
12 the source target server 44 identified by the initial secure hash/identifier pair
13 record is in fact the valid source of the encrypted policy set 200.

14 [0089] Referring again to Figure 7A, on retrieval 190 of a source encrypted
15 policy set 200 and further validation as secure and originating from a target
16 server 44 known to validly exist in the security processor cluster 18, the receiving
17 target server 44 searches the public key digest index 202 for digest value
18 matching the public key of the receiving target server 44. Preferably, the index
19 offset location of the matching digest value is used as a pointer to the data
20 structure row containing the corresponding public key encrypted triple-DES key
21 206 and triple-DES encrypted configuration policy/key data set 204. The private
22 key of the receiving target server 44 is then utilized 210 to recover the triple-DES
23 key 206 that is then used to decrypt the configuration policy/key data set 204. As
24 decrypted, the relatively updated configuration policy/key data set 204 is
25 transferred to and held in the update configuration policy/key data set memory

1 118 of the receiving target server 44. Pending installation of the updated
2 configuration policy/key data set 204, a target server 44 holding a pending
3 updated configuration policy/key data set resumes periodic issuance of cluster
4 synchronization messages 170, though using the updated configuration
5 policy/key data set version number 174.

6 [0090] In accordance with the preferred embodiments of the present
7 invention, updated configuration policy/key data sets 204 are relatively
8 synchronously installed as current configuration policy/key data sets 116 to ensure
9 that the active target servers 44_{1-Y} of the security processor cluster 18 are
10 concurrently utilizing the same version of the configuration policy/key data set.
11 Effectively synchronized installation is preferably obtained by having each target
12 server 44 wait 212 to install an updated configuration policy/key data set 204 by
13 monitoring cluster synchronization messages 170 until all such messages contain
14 the same updated configuration policy/key data set version number 174.
15 Preferably, a threshold number of cluster synchronization messages 170 must be
16 received from each active target server 44, defined as those valid target servers
17 44_{1-Z} that have issued a cluster synchronization message 170 within a defined
18 time period, for a target server 44 to conclude to install an updated configuration
19 policy/key data set. For the preferred embodiments of the present invention, the
20 threshold number of cluster synchronization messages 170 is two. From the
21 perspective of each target server 44, as soon as all known active target servers
22 44_{1-Y} are recognized as having the same version configuration policy/key data set,
23 the updated configuration policy/key data set 118 is installed 214 as the current
24 configuration policy/key data set 116. The process 160B of updating of a local
25 configuration policy/key data set is then complete 216.

1 [0091] Referring to Figure 8, an updated configuration policy/key data set
2 is generated 220 ultimately as a result of administrative changes made to any of
3 the information stored as the local configuration policy/key set data.
4 Administrative changes 222 may be made to modify access rights and similar
5 data principally considered in the policy evaluation of network requests. Changes
6 may also be made as a consequence of administrative reconfiguration 224 of the
7 security processor cluster 18, typically due to the addition or removal of a target
8 server 44. In accordance with the preferred embodiments of the present
9 invention, administrative changes 222 are made by an administrator by access
10 through the administration interface 64 on any of the target servers 44_{1-Y}. The
11 administrative changes 222, such as adding, modifying, and deleting policy rules,
12 changing encryption keys for select policy rule sets, adding and removing public
13 keys for known target servers 44, and modifying the target server 44 IP address
14 lists to be distributed to the client computers 12, when made and confirmed by the
15 administrator, are committed to the local copy of the configuration policy/key data
16 set. On committing the changes 222, the version number of the resulting updated
17 configuration policy/key data set is also automatically incremented 226. For the
18 preferred embodiments, the source encrypted configuration policy/key data set
19 200 is then regenerated 228 and held pending transfer requests from other target
20 servers 44_{1-Y}. The cluster synchronization message 170 is also preferably
21 regenerated to contain the new policy version number 174 and corresponding
22 digest set of public keys 176 for broadcast in nominal response to the local
23 heartbeat timer 162. Consequently, the newly updated configuration policy/key
24 data set will be automatically distributed and relatively synchronously installed on
25 all other active target servers 44_{1-Y} of the security processor cluster 18.

1 [0092] A reconfiguration of the security processor cluster 18 requires a
2 corresponding administrative change to the configuration policy/key data set to
3 add or remove a corresponding public key 232. In accordance with the preferred
4 embodiments of the present invention, the integrity of the security processor cluster
5 18 is preserved as against rogue or Trojan target servers 44_{1-y} by requiring the
6 addition of a public key to a configuration policy/key data set to be made only by
7 a locally authenticated system administrator or through communications with a
8 locally known valid and active target server 44 of the security processor cluster 18.
9 Specifically, cluster messages 170 from target servers 44 not already identified by
10 a corresponding public key in the installed configuration policy/key data set of a
11 receiving target server 44_{1-y} are ignored. The public key of a new target server 44
12 must be administratively entered 232 on another known and valid target server
13 44 to be, in effect, securely sponsored by that existing member of the security
14 processor cluster 18 in order for the new target server 44 to be recognized.

15 [0093] Consequently, the present invention effectively precludes a rogue
16 target server from self-identifying a new public key to enable the rogue to join the
17 security processor cluster 18. The administration interface 64 on each target
18 server 44 preferably requires a unique, secure administrative login in order to
19 make administrative changes 222, 232 to a local configuration policy/key data
20 set. An intruder attempting to install a rogue or Trojan target server 44 must have
21 both access to and specific security pass codes for an existing active target server
22 44 of the security processor cluster 18 in order to be possibly successful. Since the
23 administrative interface 64 is preferably not physically accessible from the
24 perimeter network 12, core network 18, or cluster network 46, an external breach

1 of the security over the configuration policy/key data set of the security processor
2 cluster 18 is fundamentally precluded.

3 [0094] In accordance with the preferred embodiments of the present
4 intention, the operation of the PEM components 42_{1-x} , on behalf of the host
5 computer systems 12_{1-x} , is also maintained consistent with the version of the
6 configuration policy/key data set installed on each of the target servers 44_{1-y} of
7 the security processor cluster 18. This consistency is maintained to ensure that the
8 policy evaluation of each host computer 12 network request is handled seamlessly
9 irrespective of the particular target server 44 selected to handle the request. As
10 generally shown in Figure 9, the preferred execution 240A of the PEM components
11 42_{1-x} operates to track the current configuration policy/key data set version
12 number. Generally consistent with the PEM component 42_{1-x} execution 120A,
13 following receipt of a network request 122, the last used policy version number
14 held by the PEM component 42_{1-x} is set 242 with the IP address of the selected
15 target server 44, as determined through the target server selection algorithm 128,
16 in the network request data packet. The last used policy version number is set to
17 zero, as is by default the case on initialization of the PEM component 42_{1-x} , to a
18 value based on initializing configuration data provided by a target server 44 of
19 the security processor cluster 18, or to a value developed by the PEM component
20 42_{1-x} through the cooperative interaction with the target servers 44 of the security
21 processor cluster 18. The network request data packet is then sent 130 to the
22 chosen target server 44.

23 [0095] The target server 44 process execution 240B is similarly consistent
24 with the process execution 120B nominally executed by the target servers 44_{1-y} .
25 Following receipt of the network request data packet 136, an additional check

1 244 is executed to compare the policy version number provided in the network
2 request with that of the currently installed configuration policy/key data set. If the
3 version number presented by the network request is less than the installed version
4 number, a bad version number flag is set 246 to force generation of a rejection
5 response 142 further identifying the version number mismatch as a reason for the
6 rejection. Otherwise, the network request is processed consistent with the
7 procedure 120B. Preferably, the target server process execution 240B also
8 provides the policy version number of the locally held configuration policy/key
9 data set in the request reply data packet irrespective of whether a bad version
10 number rejection response 142 is generated.

11 [0096] On receipt 144 specifically of a version number mismatch rejection
12 response, a PEM component 42_{1,x} preferably updates the network latency table
13 90 to mark 248 the corresponding target server 44 as down due to a version
14 number mismatch. Preferably, the reported policy version number is also stored
15 in the network latency table 90. A retry selection 128 of a next target server 44
16 is then performed unless 250 all target servers 44_{1,y} are then determined
17 unavailable based on the combined information stored by the security processor
18 IP address list 86 and network latency table 90. The PEM component 42_{1,x} then
19 assumes 252 the next higher policy version number as received in a bad version
20 number rejection response 142. Subsequent network requests 122 will also be
21 identified 242 with this new policy version number. The target servers 44_{1,y}
22 previously marked down due to version number mismatches are then marked up
23 254 in the network latency table 90. A new target server 44 selection is then
24 made 128 to again retry the network request utilizing the updated policy version
25 number. Consequently, each of the PEM components 42_{1,x} will consistently track

1 changes made to the configuration policy/key data set in use by the security
2 processor cluster 18 and thereby obtain consistent results independent of the
3 particular target server 44 chosen to service any particular network request.

4 [0097] Thus, a system and methods for cooperatively load-balancing a
5 cluster of servers to effectively provide a reliable, scalable network service has
6 been described. While the present invention has been described particularly with
7 reference to a host-based, policy enforcement module inter-operating with a
8 server cluster, the present invention is equally applicable to other specific
9 architectures by employing a host computer system or host proxy to distribute
10 network requests to the servers of a server cluster through cooperative
11 interoperation between the clients and individual servers. Furthermore, while the
12 server cluster service has been described as a security, encryption, and
13 compression service, the system and methods of the present invention are
14 generally applicable to server clusters providing other network services. Also,
15 while the server cluster has been describes as implementing a single, common
16 service, such is only the preferred mode of the present invention. The server
17 cluster may implement multiple independent services that are all cooperatively
18 load-balanced based on the type of network request initially received by a PEM
19 component.

20 [0098] In view of the above description of the preferred embodiments of the
21 present invention, many modifications and variations of the disclosed
22 embodiments will be readily appreciated by those of skill in the art. It is therefore
23 to be understood that, within the scope of the appended claims, the invention may
24 be practiced otherwise than as specifically described above.